Discrete Mathematics 222 (2000) 61–72

DISCRETE MATHEMATICS

Feasible edge colorings of trees with cardinality constraints

D. de Werra[a], A. Hertz[a,*], D. Kobler[a], N.V.R. Mahadev[b]

[a]*Dept de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, MA (Ecublens), CH-1015 Lausanne, Switzerland*
[b]*Northeastern University, Boston, MA, USA*

Received 13 March 1998; revised 26 October 1999; accepted 8 November 1999

Abstract

A variation of preemptive open shop scheduling corresponds to finding a feasible edge coloring in a bipartite multigraph with some requirements on the size of the different color classes. We show that for trees with fixed maximum degree, one can find in polynomial time an edge k-coloring where for $i = 1,\dots,k$ the number of edges of color i is exactly a given number h_i, and each edge e gets its color from a set $\varphi(e)$ of feasible colors, if such a coloring exists. This problem is NP-complete for general bipartite multigraphs. Applications to open shop problems with costs for using colors are described. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Edge coloring; Open shop; Cost; Timetabling; Feasible colors; Cardinality constraints

1. Introduction

In this paper, we consider an extension of the well-known edge coloring problem, which consists in determining if a color can be assigned to each edge of a graph, so that no two adjacent edges have the same color. We first impose cardinality constraints by fixing the number of times each color can be used. We then restrict the set of possible colors that each edge can receive. Determining whether a given graph has an edge-coloring satisfying the above constraints is an NP-complete problem, even if restricted to bipartite multigraphs [4]. We prove in this paper that both extensions are polynomially solvable for trees with maximum degree bounded by a constant.

The motivation for studying edge coloring problems in bipartite multigraphs stems from the classical model of preemptive open shop scheduling: we are given a collection $\mathscr{P}$ of processors $P_1,\dots,P_m$, a collection $\mathscr{J}$ of jobs $J_1,\dots,J_n$ to be processed within a period of k time units. Each job J_j consists of tasks $T_{1j},\dots,T_{mj}$; task T_{ij} of job J_j

* Corresponding author.
E-mail address: hertz@dma.epfl.ch (A. Hertz).

has to be processed on processor P_i. Its processing time p_{ij} is given and we assume that it is integral. If $p_{ij} = 0$, then T_{ij} does not exist. No processor can work on two tasks simultaneously and no two tasks of the same job can be processed at the same time. The tasks of the same job can be processed in any order. Furthermore we assume that preemptions are allowed (after any integral number of time units) during the processing of a task on a processor. The *preemptive open shop scheduling problem* (OSSP) consists in scheduling all jobs within k time units while satisfying all requirements described above.

A well-known application of this model is the simple class–teacher timetabling problem: each P_i is a teacher and each J_j a class, i.e., a group of students following exactly the same program. Then T_{ij} is the collection of p_{ij} lectures (of one time unit) that teacher P_i has to give to class J_j. We associate with this problem a bipartite multigraph $G=(\mathscr{P},\mathscr{J},E)$ constructed as follows: each P_i corresponds to a node in the left set $\mathscr{P}$ of nodes and each J_j to a node in the right set $\mathscr{J}$ of nodes. P_i and J_j are linked by p_{ij} parallel edges.

An *edge k-coloring* is an assignment F of one color $F(e)$ in $\{1,2,\dots,k\}$ to each edge e of $G=(\mathscr{P},\mathscr{J},E)$ such that $F(e)\neq F(g)$ whenever edges e and g are adjacent (i.e., share at least one node). Edge k-colorings and feasible schedules for OSSP are in correspondence: $F([P_i,J_j])=l$ means that task T_{ij} is in process during the lth time unit and that P_i gives a lecture to class J_j at that time. If $\Delta(G)$ is the maximum degree of G (the maximum number of edges adjacent to the same node), then it is known that an edge k-coloring of G exists if and only if $k\geqslant\Delta(G)$ (see [1]). All graph–theoretical terms not defined here can be found in [1].

Many variations and extensions of the OSSP have been studied (see [2,8,16–18]) for dealing with special types of scheduling or timetabling problems. One such situation arises when in a timetabling problem each lecture associated to an edge e must be scheduled to one time unit in a set $\varphi(e)$ of feasible time units. We have then a problem of *restricted edge coloring* (G,ϕ): given a bipartite multigraph G and a family ϕ of sets $\varphi(e)$ of feasible colors for each edge e, find an edge coloring of G such that each edge gets a feasible color. This problem is generally NP-complete [4] but solutions can be found in polynomial time if G is a forest [17].

Another variation motivated by applications in timetabling consists in introducing a cost $c(i,e)$ incurred when edge e gets color i. Such a cost may translate the degree of preference given to the assignment of lecture e to a time unit i. Finding an edge k-coloring with minimum cost may give a timetable in which one tries to avoid as much as possible the undesired assignment of lectures (for instance, by setting $c(i,e)=\infty$, we simply express that $i\notin\varphi(e)$). Other applications to VLSI and to machine scheduling are described in [12].

An edge k-coloring S of G will be denoted by $S=(M_1,M_2,\dots,M_k)$: M_i is the matching consisting of all edges with color i. The cost $f(S)$ of S is then given by

$$f(S)=\sum_{i=1}^{k}\sum_{e\in M_i}c(i,e). \tag{1.1}$$

Finding an edge k-coloring with minimum cost in a bipartite multigraph is generally NP-complete (see [10,11]); when G is a forest, a polynomial algorithm based on dynamic programming gives a coloring with minimum cost (when the maximum degree k of G is fixed) [14].

In some cases, we may assume that the costs $c(i,e)$ do not depend upon the edges, so that we have $c(i,e)=c_i$ for each edge and each color i. Eq. (1.1) then becomes

$$f(S)=\sum_{i=1}^{k} c_i|M_i|, \tag{1.2}$$

where $|M_i|$ denotes the cardinality of the set M_i of edges. So the cost of a coloring will depend only upon the sequence $(|M_1|,|M_2|,\ldots,|M_k|)$ of cardinalities of the different color classes.

It is therefore appropriate to recall a few properties of the sequences $(h_1,h_2,\ldots,h_k)$ representing the cardinalities of the color classes in an edge k-coloring of a graph G. This will be done in the next section. We characterize the sequences $(h_1,h_2,\ldots,h_k)$ for which there exists an edge k-coloring $S=(M_1,M_2,\ldots,M_k)$ with $|M_i|=h_i$ $(i=1,\ldots,k)$, and such that S minimizes function f in Eq. (1.2). In Section 3, we show how to generate all such sequences in a tree with fixed maximum degree. In Section 4, we extend this algorithm to the case where the set of possible colors of each edge is restricted. A conclusion follows in Section 5.

2. Some properties of edge colorings

Given a multigraph $G=(V,E)$ without loops, a sequence $H=(h_1,\ldots,h_k)$ of integers with $h_1\geqslant\cdots\geqslant h_k$ and $\sum_{i=1}^{k} h_i=|E|$ will be called *color-feasible* (for G) if there exists an edge k-coloring $S=(M_1,\ldots,M_k)$ of G with $|M_i|=h_i$ for $i=1,\ldots,k$.

For two sequences $H=(h_1,\ldots,h_k)$ with $h_1\geqslant\cdots\geqslant h_k$ and $H'=(h'_1,\ldots,h'_k)$ with $h'_1\geqslant\cdots\geqslant h'_k$ we shall write $H\geqslant H'$ if

$$\sum_{i=1}^{r}(h_i-h'_i)\geqslant 0 \quad (r=1,\ldots,k) \text{ with equality holding for } r=k. \tag{2.1}$$

Observe that we allow h_i to be zero. Moreover, relation '$\geqslant$' induces a partial order on the set of sequences of given length k. Let $C(G)$ be the set of all color-feasible sequences (note that we may take $k=|E|$ in order to have a finite set $C(G)$).

Property 2.1 (Folkman and Fulkerson [5]). *If $H\in C(G)$ and $H\geqslant H'$, then $H'\in C(G)$.*

A sequence $H\in C(G)$ is called *maximal* if there is no $H'\in C(G)$ with $H'\neq H$ and $H'\geqslant H$. Maximal color-feasible sequences of bipartite multigraphs have the following interesting property.

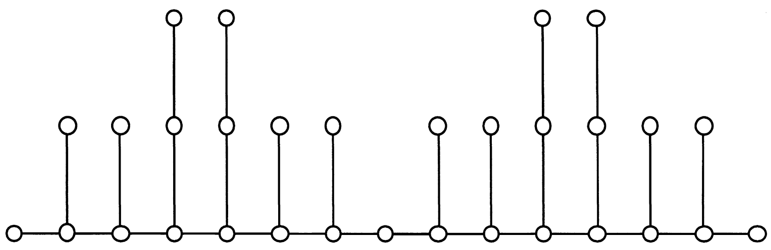

Fig. 1. A tree G with maximal sequences $H_1=(14,8,8)$, $H_2=(12,12,6)$ and $H_3=(13,10,7)=\frac{1}{2}(H_1+H_2)$.

Property 2.2 (Folkman and Fulkerson [5]; de Werra [13]). *If G is a bipartite multigraph, then all maximal sequences in $C(G)$ have exactly $\Delta(G)$ positive members.*

When costs c_i are introduced, we may assume $c_1 \leqslant c_2 \leqslant \cdots \leqslant c_k$, without loss of generality. For minimizing the cost $f(S)$ of a schedule (or of an edge k-coloring) $S=(M_1,\ldots,M_k)$ we only have to find in $C(G)$ a sequence $H=(h_1,\ldots,h_k)$ for which $\sum_{i=1}^{k} c_i h_i$ is minimum.

Property 2.3 (de Werra et al. [16]). *The minimum value of $\sum_{i=1}^{k} c_i h_i$ where $H=(h_1,\ldots,h_k)\in C(G)$ is obtained for a maximal sequence H.*

In order to solve this problem one may think of generating all maximal sequences in $C(G)$; this is a difficult problem when G is a bipartite multigraph [6]. Some special cases where the minimization of $f(S)$ can be solved in polynomial time are described in [16].

As a consequence of Property 2.1, we notice that if we know all maximal sequences in $C(G)$, then we can easily generate the remaining sequences in $C(G)$. So the knowledge of all maximal sequences may also be needed for the situation where the cost function is not linear and Property 2.3 does not necessarily hold.

Let us now restrict our attention to trees (or forests); is it possible to generate all maximal sequences of $C(G)$ in polynomial time? A natural idea would be to use the dynamic programming algorithm of de Werra [14] in order to find an edge k-coloring S which minimizes the cost $f(S)$ given by (1.2). By varying the costs, one could hope to generate all maximal sequences. However this approach would fail to give all maximal sequences of $C(G)$ (if the algorithm provides just one optimal solution for each set of costs $c_1,\ldots,c_k$). Indeed, one may construct a tree G with $\Delta(G)=3$ and for which the maximal sequences in $C(G)$ are $H_1=(14,8,8)$, $H_2=(12,12,6), H_3=(13,10,7)$ (see Fig. 1). Clearly since $H_3=\frac{1}{2}(H_1+H_2)$, it will never be the unique optimal solution of a minimum cost problem with $f(S)$ given by (1.2). So we may not be certain to generate it. It is therefore appropriate to develop another type of algorithm generating all maximal sequences in $C(G)$ when G is a tree. This will be described in the next section.

3. Dynamic programming procedure

Given a graph G, determining whether a given sequence is color-feasible for G is an NP-complete problem, even when restricted to the class of bipartite multigraphs with maximum degree at most three [4]. We show below that the problem is polynomially solvable for trees with bounded degrees using a dynamic programming method similar to the one used in [9] for a special type of node coloring of trees.

While the bipartite multigraphs used to model preemptive open shop scheduling problems are rarely trees, the above result is interesting for at least two reasons. There is first a theoretical interest since we answer an open question. Moreover, we hope that this polynomial algorithm will help in the design of efficient heuristic algorithms for general bipartite multigraphs.

The MAXIMAL COLOR-FEASIBLE SEQUENCES problem is defined as follows:

Input: Tree $T=(V,E)$ with $\Delta(T)\leqslant K$ (fixed)

Problem: Compute all the maximal color-feasible sequences of T.

To solve this problem, we first orient the edges of T towards a leaf chosen as root. For each arc (oriented edge) (x,y), x is called the *tail* and y the *head*. An arc (u,x) is called a *child* of (x,y) and an arc (y,v) is called a *parent* of (x,y). An arc whose tail is a leaf is called a *stock* and the arc whose head is the root is called the *root-arc*. Clearly every arc other than the root-arc has a single parent, and the root-arc has none.

Label the arcs of T as $e_1,\dots,e_{|E|}$ so that if e_i is a child of e_j then $i<j$. Let T_e be the maximal subtree of T with e as the root-arc.

We need to consider the particular color assigned to each edge, so let $1,2,\dots,k$ be the colors used in any edge k-coloring of T. According to Property 2.2 we can assume that $k=\Delta(T)$.

Let $d_1,d_2,\dots,d_{k-2}$ be a sequence of non-negative integers with $\sum_{j=1}^{k-2} d_j\leqslant|E|$. Consider any arc e and its corresponding subtree T_e. Let $N(e,j;d_1,\dots,d_{k-2})$ be the maximum number of arcs in T_e that can have color $k-1$ in a k-coloring of T_e, provided that

- exactly d_i arcs have color i $(1\leqslant i\leqslant k-2)$
- arc e has a given color j $(1\leqslant j\leqslant k)$.

Note that it may not be defined if no such k-coloring exists.

We compute the list of all such numbers at each arc by processing in the order $e_1,\dots,e_{|E|}$. Note that at the start of processing arc e_i, the list at each of its children has been computed.

Let e_i be any parent arc with children $e_{i_1},\dots,e_{i_d}$. In order to compute the list of numbers $N(e_i,j;d_1,\dots,d_{k-2})$ for T_{e_i}, we consider the subtrees $T_{e_{i_1}}$, $T_{e_{i_1}}\cup T_{e_{i_2}},\dots,\bigcup_{s=1}^{d} T_{e_{i_s}}$. So let $T_{e_i}^r=\bigcup_{s=1}^{r} T_{e_{i_s}} (1\leqslant r\leqslant d)$ and consider any Boolean vector $\gamma=(\gamma_1,\dots,\gamma_k)$ in $\mathbb{B}^k$ with exactly r non-zero components. Let $N'(e_i,r;\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})$ be the maximum number of arcs in $T_{e_i}^r$ which can receive color $k-1$, provided that

- color j $(1\leqslant j\leqslant k)$ appears on one of the arcs $e_{i_1},\dots,e_{i_r}$ if and only if $\gamma_j=1$,
- exactly d_j arcs $(1\leqslant j\leqslant k-2)$ in $T^r_{e_i}$ have color j.

The following procedure, called LIST, generates the list $C_{\max}(T)$ of all maximal color-feasible sequences of a tree T.

```
Algorithm LIST (T)
(a) Orient the edges of $T$ towards a leaf chosen as root. Label the arcs of $T$ as
    $e_1,\dots,e_{|E|}$ so that if $e_i$ is a child of $e_j$ then $i<j$.
(b) Computation of the numbers $N$
    For $i=1$ to $|E|$ do
     (b1) If $e_i$ is a stock then
          set $N(e_i,j;d_1,d_2,\dots,d_{k-2})=\begin{cases}0 & \text{if } j<k-1, d_j=1 \text{ and } d_r=0 \text{ for } r\neq j\\ 1 & \text{if } j=k-1 \text{ and } d_1=\cdots=d_{k-2}=0\\ 0 & \text{if } j=k, \text{ and } d_1=\cdots=d_{k-2}=0\\ \text{undefined} & \text{otherwise}\end{cases}$
     (b2) Otherwise $e_i$ is a parent. Let $e_{i_1},\dots,e_{i_d}$ be all the children of $e_i$.
          (b2.1) For each sequence $(j;d_1,\dots,d_{k-2})$ such that $N(e_{i_1},j;d_1,\dots,d_{k-2})$ is
                 defined do
                   set $N'(e_i,1;\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})=N(e_{i_1},j;d_1,\dots,d_{k-2})$ with
                   $\gamma_j=1$ and $\gamma_t=0\ \forall t\neq j$
          (b2.2) For $r=2$ to $d$ do
                   For each sequence $(\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})$ such that $N'(e_i,r-1;$
                      $\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})$ is defined do
                     For each sequence $(j;d'_1,\dots,d'_{k-2})$ with $\gamma_j=0$, such that
                        $N(e_{i_r},j;d'_1,\dots,d'_{k-2})$ is defined do
                      set $N'(e_i,r;\gamma_1,\dots,\gamma_{j-1},1,\gamma_{j+1},\dots,\gamma_k;d_1+d'_1,\dots,d_{k-2}+d'_{k-2})$
                        $=N'(e_i,r-1;\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})+N(e_{i_r},j;d'_1,\dots,d'_{k-2})$
                      unless the left-hand-side value is already larger than the
                      right-hand-side.
          (b2.3) For each sequence $(\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})$ such that $N'(e_i,d;$
                 $\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})$ is defined do
                   For each $j$ such that $\gamma_j=0$ do
                    set $x=\begin{cases}N'(e_i,d;\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2})+1 & \text{if } j=k-1\\ N'(e_i,d;\gamma_1,\dots,\gamma_k;d_1,\dots,d_{k-2}) & \text{otherwise}\end{cases}$
                    set $(d'_1,\dots,d'_{k-2})=\begin{cases}(d_1,\dots,d_{k-2}) & \text{if } j\geqslant k-1\\ d'_j=d_j+1, d'_t=d_t\ \forall t\neq j & \text{otherwise}\end{cases}$
                    set $N(e_i,j;d'_1,\dots,d'_{k-2})=x$ unless the left-hand-side value
                      is already larger than the right-hand-side.
      End otherwise
```

(c) *Construction of the list $C_{\max}(T)$ of maximal color-feasible sequences*
Set $C_{\max}(T) = \{(d_1,\dots,d_k) | \sum_{t=1}^{k} d_t = |E|$ and $\exists j$, $1 \leqslant j \leqslant k$, with $d_{k-1} = N(e_{|E|}, j; d_1,\dots,d_{k-2})\}$. Remove from $C_{\max}(T)$ any sequence whose members are not in non-increasing order.
Remove from $C_{\max}(T)$ all sequences H' such that $\exists H \in C_{\max}(T)$ with $H \neq H'$ and $H \geqslant H'$.

End of LIST

Property 3.1. *Given a tree T with maximum degree bounded by a constant, the LIST algorithm generates all maximal color-feasible sequences in polynomial time.*

Proof.

(1) *Every maximal feasible sequence is in $C_{\max}(T)$.* Since no maximal feasible sequence is removed from $C_{\max}(T)$ in Step (c), it is sufficient to prove that for each maximal color-feasible sequence $(d_1,\dots,d_k)$, there exists a j, $1 \leqslant j \leqslant k$, with $d_{k-1} = N(e_{|E|}, j; d_1,\dots,d_{k-2})$. Consider any feasible coloring of T in which exactly d_t arcs have color t $(1 \leqslant t \leqslant k)$, and let j be the color of the root-arc $e_{|E|}$. It follows from the definition that $N(e_{|E|}, j; d_1,\dots,d_{k-2}) \geqslant d_{k-1}$. If the inequality is strict, then $(d_1,\dots,d_k)$ is not a maximal sequence, a contradiction.

(2) *LIST is a polynomial time algorithm.*

(a) Step (a) can easily be performed in $O(|E|)$ time.

(b1) For a stock $e_i, N(e_i, j; d_1, d_2,\dots,d_{k-2})$ can be computed in $O(k)$ time.

(b2.1) There are at most $k|E|^{k-2}$ sequences $(j; d_1,\dots,d_{k-2})$. Hence, this step can be performed in $O(k|E|^{k-2})$.

(b2.2) There are at most $C_{r-1}^{k}|E|^{k-2}$ sequences $(\gamma_1,\dots,\gamma_k; d_1,\dots,d_{k-2})$ for which $N'(e_i, r-1; \gamma_1,\dots,\gamma_k; d_1,\dots,d_{k-2})$ is defined, and at most $(k-r+1)|E|^{k-2}$ sequences $(j; d'_1,\dots,d'_{k-2})$ with $\gamma_j = 0$ and for which $N(e_{i_r}, j; d'_1,\dots,d'_{k-2})$ is defined. Hence, the list of numbers $N'(e_i, d; \gamma_1,\dots,\gamma_k; d_1,\dots,d_{k-2})$ for e_i can be computed in $O(d(k+1)!|E|^{2k-4})$ time.

(b2.3) There are at most $C_d^k|E|^{k-2}$ sequences $(\gamma_1,\dots,\gamma_k; d_1,\dots,d_{k-2})$ for which $N'(e_i, d; \gamma_1,\dots,\gamma_k; d_1,\dots,d_{k-2})$ is defined, and at most $(k-d)$ colors j with $\gamma_j = 0$. Hence, the list of numbers $N(e_i, j; d_1, d_2,\dots,d_{k-2})$ for e_i can be computed in $O((k+1)!|E|^{k-2})$ time.

(b) Since steps (b1) and (b2) are performed $|E|$ times, and since each arc in T has at most $\Delta(T) - 1$ children, the computation of all numbers $N(e_i, j; d_1, d_2,\dots, d_{k-2})$ takes $O(\Delta(T)(k+1)!|E|^{2k-3})$ time.
We know from Property 2.2. that all maximal color-feasible sequences have exactly $k = \Delta(T)$ positive members. But $\Delta(T)$ is bounded by a given value K. It follows that Step (b) can be performed in $O(|E|^{2K-3})$ time.

(c) For the root-arc e, there are at most $k|E|^{k-2}$ sequences $(j, d_1, d_2, \ldots, d_{k-2})$ for which $N(e, j; d_1, d_2, \ldots, d_{k-2})$ is defined. Hence, computing $C_{\max}(T)$ takes $O(|E|^{k-2})$ time. Removing sequences whose members are not in non-increasing order can also be done in $O(|E|^{k-2})$ and the number of comparisons to be made to eliminate dominated sequences is in $O((|E|^{k-2})^2)$. Thus, the total complexity of (c) belongs to $O(|E|^{2K-4})$.

It follows that the LIST algorithm computes all maximal color-feasible sequences in $O(|E|^{2K-3})$ time. □

Corollary. Given a tree T with maximum degree bounded by a constant, it can be decided in polynomial time whether a given sequence is color-feasible for T.

Proof. If follows from Property 2.1 that a sequence H is color-feasible for T if and only if there exists a maximal color-feasible sequence H' of T such that $H' \geqslant H$. Since, the LIST algorithm computes all maximal color-feasible sequences of T in polynomial time, this decision problem is polynomially solvable. □

4. Additional constraints

In practically all timetabling problems several types of constraints are present. Consider the restricted edge coloring problem described in Section 1, with *cardinality constraints* on the edge coloring $S = (M_1, M_2, \ldots, M_k)$: for each i the number $|M_i|$ of edges which have color i is h_i. Let (G, ϕ, H) be the restricted edge k-coloring problem with cardinality constraints defined by a sequence $H = (h_1, \ldots, h_k)$.

Property 4.1 (Dror et al. [3]). *(G, ϕ, H) is NP-complete even if G is restricted to a chain and $|\varphi(e)| = 2$ for each edge e.*

Property 4.2 (de Warra [14]). *If G is a collection of node-disjoint stars, then (G, ϕ, H) can be solved in polynomial time.*

In fact the line-graph $L(G)$ of G is in the latter case a collection of node-disjoint cliques; the problem in $L(G)$ becomes a node coloring problem and a model based on network flows is given in [14]. It produces either a node-coloring of $L(G)$ or a proof of non existence. We recall the construction of the network N in [14]: introduce a source s, a sink t, a node x for each node of $L(G)$, nodes (K_i, j) for each clique K_i of $L(G)$ and for each color j in $\{1, \ldots, k\}$ and a node j for each color j. The arcs are constructed as in Table 1.

There exists a node k-coloring of $L(G)$ satisfying all requirements if and only if there exists an integral compatible flow from s to t in N with value equal to the

Table 1

	Arcs (u, v)	Capacities $c(u, v)$	Lower bounds $l(u, v)$
(s, v)	if $v \in V$	1	0
$(v, (K_i, j))$	if $v \in K_i$ and $j \in \varphi(v)$	∞	0
$((K_i, j), j)$	$i = 1, \dots, p$	1	0
(j, t)	$j = 1, \dots, k$	h_j	0

number of edges in G. Necessary and sufficient conditions have been derived in [14] for the existence of a node k-coloring of $L(G)$.

Notice that we may introduce costs c_j on the arcs (j, t) of N, so that by constructing a maximum flow from s to t in N with minimum cost, we will get an edge k-coloring S in G which minimizes the cost

$$f(S) = \sum_{i=1}^{k} c_i |M_i|.$$

Since we are using a network formulation, we may as well try to obtain a restricted coloring $S = (M_1, \dots, M_k)$ which in addition to the minimum cost requirement may satisfy the following requirements:

$$\alpha_1 \leqslant |M_1| \leqslant \beta_1 \equiv h_1,$$

$$\alpha_2 \leqslant |M_1| + |M_2| \leqslant \beta_2,$$

$$\alpha_{k-1} \leqslant |M_1| + |M_2| + \cdots + |M_{k-1}| \leqslant \beta_{k-1}.$$

This is obtained by a simple modification of the network. Similarly restricted colorings with *nested* constraints have been considered in [15].

Note that if there are no cardinality constraints, (G, ϕ, H) is solvable in polynomial time for trees (with the algorithm in [17]). Similarly when $\varphi(e) = \{1, \dots, K\}$ with a fixed K for each edge e, then (G, ϕ, H) is solvable in polynomial time for trees (with the algorithm of Section 3).

More generally if $\varphi(e) \subseteq \{1, \dots, K\}$ with a fixed K, then (G, ϕ, H) is solvable in polynomial time for trees with an adaptation of the LIST algorithm in Section 3. More precisely, consider the RESTRICTED COLOR-FEASIBLE SEQUENCES problem defined as follows.

Input: Tree $T = (V, E)$ with a list $\varphi(e) \subseteq \{1, \dots, K\}$ (K fixed) of allowed colors for each $e \in E$

Problem: Compute the set $C(T)$ containing all sequences $(h_1, \dots, h_K)$ of T such that there is a restricted edge coloring of T with exactly h_i edges of color i.

Note that we may assume that $\Delta(T) \leqslant K$, since otherwise $C(T) = \emptyset$. As in Section 3, we first orient the edges of T towards a leaf chosen as root, and then label the arcs of T as $e_1, \dots, e_{|E|}$ so that if e_i is a child of e_j then $i < j$.

Let e be any arc of T. The sequence of integers $(j;d_1,\dots,d_K)$ is said to be a feasible color-mapping of T_e if there exists a restricted edge coloring of T_e where e has color j and there are d_t arcs of color t $(1\leqslant t\leqslant K)$ in T_e. We denote by F_e the set of the feasible color-mappings of T_e. We will recursively compute the sets $F_{e_1},F_{e_2},\dots,F_{e_{|E|}}$.

Let e_i be any parent arc with children $e_{i_1},\dots,e_{i_d}$. In order to compute the sets F_{e_i}, we again consider the subtrees $T^r_{e_i}$ $1\leqslant r\leqslant d$ and Boolean vectors $(\gamma_1,\dots,\gamma_K)$ with exactly r non-zero components. A sequence $(\gamma_1,\dots,\gamma_K;d_1,\dots,d_k)$ is said to be a feasible color-mapping of $T^r_{e_i}$ if there exists a restricted edge coloring of $T^r_{e_i}$ such that

- color t $(1\leqslant t\leqslant K)$ appears on one of the arcs $e_{i_1},\dots,e_{i_r}$ if and only if $\gamma_t=1$,
- exactly d_t arcs in $T^r_{e_i}$ have color t $(1\leqslant t\leqslant K)$.

The set of feasible color-mappings of $T^r_{e_i}$ will be denoted by $F^r_{e_i}$. The algorithm that generates the set $C(T)$ can be described as follows:

```
*Algorithm LIST-restricted* ($T$)
  (a) Orient the edges of $T$ towards a leaf chosen as root. Label the arcs of $T$ as
      $e_1,\dots,e_{|E|}$ so that if $e_i$ is a child of $e_j$ then $i<j$.
  (b) **For** $i=1$ **to** $|E|$ **do**
      (b1) **If** $e_i$ is a stock **then**
               $F_{e_i}=\{(j;d_1,\dots,d_K)\mid j\in\varphi(e_i),d_j=1,d_t=0\ \forall t\neq j\}$
      (b2) **Otherwise** $e_i$ is a parent. Let $e_{i_1},\dots,e_{i_d}$ be all the children of $e_i$
          (b2.1) $F^1_{e_i}=\{(\gamma_1,\dots,\gamma_k;\ d_1,\dots,d_K)\mid(j;\ d_1,\dots,d_K)\in F_{e_{i_1}},\gamma_j=1,$
                 $\gamma_t=0\ \forall t\neq j\}$
          (b2.2) **For** $r=2$ **to** $d$ **do**
                    $F^r_{e_i}=\emptyset$
                    **For** each $(\gamma_1,\dots,\gamma_K;d_1,\dots,d_K)\in F^{r-1}_{e_i}$ **do**
                       **For** each $(j;d'_1,\dots,d'_K)\in F_{e_{i_r}}$ such that $\gamma_j=0$ **do**
                          Put$(\gamma_1,\dots,\gamma_{j-1},1,\gamma_{j+1},\dots,\gamma_k;d_1+d'_1,\dots,d_K+d'_K)$in$F^r_{e_i}$.
          (b2.3) $F_{e_i}=\emptyset$
                 **For** each $(\gamma_1,\dots,\gamma_K;d_1,\dots,d_K)\in F^d_{e_i}$ **do**
                    **For** each $j\in\varphi(e_i)$ such that $\gamma_j=0$ **do**
                       put $(j;d_1,\dots,d_{j-1},d_{j+l},d_{j+1},\dots,d_K)$ in $F_{e_i}$.
      **End otherwise**
  (c) $C(T)=\{(d_1,\dots,d_K)\mid\exists j,\ 1\leqslant j\leqslant K,$ with $(j;d_1,\dots,d_K)\in F_{e_{|E|}}\}$
**End of LIST-restricted**
```

Property 4.1. *The* LIST*-restricted algorithm is a polynomial time algorithm.*

Proof.

(a) Step (a) can easily be performed in $O(|E|)$ time.

(b1) For a stock e_i,F_{e_i} can be generated in $O(K)$ time.

(b2.1) Notice that $|F_e| \leqslant K(|E|+1)^{K-1}$ for each edge e in E. Therefore, step (b2.1) can be performed in $O(K|E|^{K-1})$ time.

(b2.2) If $(\gamma_1,\dots,\gamma_K; d_1,\dots,d_K) \in F_{e_i}^{r-1}$ then $\gamma = (\gamma_1,\dots,\gamma_K)$ has exactly $(r-1)$ non-zero components. Hence, $|F_{e_i}^{r-1}| \leqslant C_{r-1}^{K}(|E|+1)^{K-1}$ and it follows that step (b2.2) can be performed in $O(\sum_{r=2}^{d}(C_{r-1}^{K}|E|^{K-1})((K-r+1)|E|^{K-1}))$. Since K is a constant and since $d < \Delta(T) \leqslant K$, we conclude that step (b2.2) takes $O(|E|^{2K-2})$ time.

(b2.3) This step can be performed in $O((K-d)C_d^K|E|^{K-1}) = O(|E|^{K-1})$ (since K is fixed).

(b) Since steps (b1) and (b2) are performed $|E|$ times, the computation of all sets F_{e_i} takes $O(|E|^{2K-1})$ time.

(c) Computing $C(T)$ takes $O(K|E|^{K-1})$ time.

It follows that the LIST-restricted algorithm runs in $O(|E|^{2K-1})$ time. $\square$

The above algorithm solves in fact a more general problem than the algorithm in Section 3: we have here introduced sets $\varphi(e)$ of feasible colors for each edge e. In such a situation, $C(T)$ no longer satisfies Property 2.1; so we need to generate all feasible sequences separately. The algorithm in Section 3 eliminates at each step subsequences which may not be extended to maximal sequences. This can be done when there is no restriction on the colors to be used. Indeed, according to Property 2.3, only maximal sequences have to be generated in that case. Property 2.3 is however not true in the case when we have arbitrary sets of feasible colors for the edges. Note that the LIST algorithm has a lower complexity when compared to the LIST-restricted algorithm.

5. Concluding remarks

We have given here a polynomially solvable case of edge coloring where feasible sets of colors are given and cardinality constraints must be satisfied.

Many related results on coloring with costs can be found in [10,11]. It is known from [7] that the restricted node K-coloring problem with cardinality constraints on a tree is polynomial when the number K of colors is given. However some questions are still open. For example, our knowledge of the set $C(G)$ of color-feasible sequences of a bipartite multigraph G is extremely fragmentary; even for trees we know very little; bounds on the number of maximal sequences would be useful for developing enumerative schemes.

Acknowledgements

This paper was written while the fourth author was visiting the Department of Mathematics of the Ecole Polytechnique Fédérale de Lausanne in September 1997; the support of EPFL is gratefully acknowledged.

References

[1] C. Berge, Graphs and Hypergraphs, North-Holland, Amsterdam, 1973.
[2] J. Blazewicz, K. Ecker, G. Schmidt, J. Weglarz, Scheduling in Computer and Manufacturing Systems, Springer, Berlin, 1993.
[3] M. Dror, G. Finke, S. Gravier, W. Kubiak, On the complexity of a restricted list-coloring problem, Discrete Math. 195 (1999) 103–109.
[4] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM J. Comput. 5 (1976) 691–703.
[5] J. Folkman, D.R. Fulkerson, Edge colorings in bipartite graphs, in: R.C. Bose and T.A. Dowling (Eds.), Combinatorial Mathematics and its Applications, University of North Carolina Press, Chapel Hill, 1969, pp. 561–577.
[6] H. Gabow, T. Nishizeki, O. Kariv, D. Leven, O. Terada, Algorithms for edge-coloring graphs, unpublished manuscript, University of Colorado, Boulder, 1983.
[7] S. Gravier, D. Kobler, W. Kubiak, Complexity of list coloring problems with a fixed number of colors, ORWP-EPFL 97/13, September 1997.
[8] P. Hansen, A. Hertz, J. Kuplinsky, Bounded vertex colorings of graphs, Discrete Math. 111 (1993) 305–312.
[9] P. Hansen, A. Hertz, N. Quinodoz, Splitting trees, Discrete Math. 165/166 (1997) 403–419.
[10] K. Jansen, Complexity results for the optimum cost chromatic partition problem, Report 96-41, Universität Trier, 1996.
[11] K. Jansen, Approximation results for the optimum cost chromatic partition problem, Report 97-01, Universität Trier, 1997.
[12] S. Nicoloso, M. Sarrafzadeh, X. Song, On the sum coloring problem on interval graphs, R. 390, IASI-CNR, Roma, 1994.
[13] D. de Werra, Some remarks on good colorations, J. Combin. Theory 21 (1976) 57–64.
[14] D. de Werra, Restricted coloring models for timetabling, Discrete Math. 165/166 (1997) 161–170.
[15] D. de Werra, On a Multiconstrained Model for Chromatic Scheduling, Discrete Appl. Math. 94 (1999) 171–180.
[16] D. de Werra, F. Glover, E.A. Silver, A chromatic scheduling model with costs, IIE Trans. 27 (1995) 181–189.
[17] D. de Werra, A. Hoffman, N.V.R. Mahadev, U. Peled, Restrictions and Preassignments in Preemptive Open Shop Scheduling, Discrete Appl. Math. 68 (1996) 169–188.
[18] D. de Werra, N.V.R. Mahadev, Preassignment requirements in chromatic scheduling, Discrete Appl. Math. 76 (1997) 95–101.